ADVANCED ALGORITHM

LAB-01
RANDOMIZED QUICKSORT
CODE:

```
#include <iostream>
using namespace std;
int count=0;
int partition(int A[],int p,int r)
{
   int x = A[r];
   int i=p-1;
   int j;

   for(j=p;j<=r-1;j++)
   {
      count++;
      if(A[j]<=x)
      {
         i++;
         swap(A[i],A[j]);

      }
   }
   swap(A[i+1],A[r]);
   return i+1;

}
int randomized_partition(int A[],int p,int r)
{

      int a= p+rand()%(r-p+1);
      swap(A[a],A[r]);

      return partition(A,p,r);
}
int randomized_quicksort(int A[],int p,int r)
{
   if(p<r)
   {
      int q=randomized_partition(A,p,r);
      randomized_quicksort(A,p,q-1);
      randomized_quicksort(A,q+1,r);
   }
```

```
   return 0;
}
int main()
{
    srand(time(NULL));
    int A[1002];
    for(int i=0;i<=1000;i++)
    {
       A[i] = i;
    }
   randomized_quicksort(A,1,1000);
   cout<<"No of Comparison:"<<count;
}
```

Kth SMALLEST ELEMENT

```
#include<iostream>
#include<climits>
#include<cstdlib>
using namespace std;

int randomPartition(int arr[], int l, int r);
int kthSmallest(int arr[], int l, int r, int k)
{
        if (k > 0 && k <= r-l+1)
        {
                int pos = randomPartition(arr, l, r);
                if (pos-l == k-1)
                        return arr[pos];
                if (pos-l > k-1)
                        return kthSmallest(arr, l, pos-1, k);
                return kthSmallest(arr, pos+1, r, k-pos+l-1);
        }

        return INT_MAX;
}
int partition(int arr[], int l, int r)
{
        int x = arr[r], i = l;
        for (int j = l; j <= r - 1; j++)
        {
                if (arr[j] <= x)
                {
```

```
                        swap(arr[i], arr[j]);
                        i++;
                }
        }
        swap(arr[i],arr[r]);
        return i;
}
int randomPartition(int arr[], int l, int r)
{
        int n = r-l+1;
        int pivot = rand() % n;
        swap(arr[l + pivot], arr[r]);
        return partition(arr, l, r);
}
int main()
{
        int arr[] = {1,8,5,3,11,12,6,97,33};
        int n = sizeof(arr)/sizeof(arr[0]), k = 4;
        cout << "K'th smallest element is " << kthSmallest(arr, 0, n-1, k);
        return 0;
}
```

LAB=02
PRIMALITY TESTING

```
#include <iostream>
using namespace std;
float count=0;
int gcd(int a,int b)
{
   if(a<b)
   {
      return gcd(b,a);
   }
   else if(a%b==0)
   {
      return b;
   }
   else
   {
      return gcd(b,a%b);
   }
}
int power(int a,unsigned int x,int p)
{
   int res=1;
   a = a%p;
   while(x>0)
   {
      if(x & 1)
      {
         res = (res*a)%p;
      }
      x=x/2;
      a=(a*a)%p;
   }

   return res;
}
bool isPrime(unsigned long int n,int k)
{
   if(n<=1 || n==4)
   {
      return false;
```

```
    }
    if(n<=3)
    {
        return true;
    }
    while(k>0)
    {
        int a = 2 + rand() % (n-4);
        if(gcd(n,a)!=1)
        {
            return false;
        }
        if(power(a,n-1,n)!=1)
        {
            return false;
        }
        for(a=2;a<=n;a++)
        {
            int r=power(a,n-1,n);
            if(r==1)
            {
                count++;
            }
        }
        k--;
    }


     return true;
}
int main()
{
    int k=3;
    if(isPrime(1009,k)==true)
    {
        cout<<"Prime";
    }
    else
    {
        cout<<"Composite";
    }
     cout<<endl<<"Count: "<<count;
    return 0;
}
```

Output

Clear

```
/tmp/Lp8OOTRf2E.o
Prime
Count: 3021
```